Indoor Air Quality Monitoring System

Abdullah Khan
Department Of Computer Science
Santa Clara University
Santa Clara, California
amkhan@scu.edu

Bharti Prakash
Department Of Computer Science
Santa Clara University
Santa Clara, California
bprakash2@scu.edu

Kushali Harish
Department Of Computer Science
Santa Clara University
Santa Clara, California
kharish@scu.edu

Air quality monitoring system is a process used to determine existing quality of air, evaluate effectiveness of control programs and to identify areas in need of restoration. Air quality monitoring systems are essential tools for assessing the quality of the air we breathe. These systems measure the concentration of various pollutants in the air, such as particulate matter, nitrogen oxides, and hydrogen. They can be used to detect and track changes in air quality over time and to identify sources of pollution. Air quality monitoring systems are important for protecting public health, as exposure to polluted air can lead to respiratory problems, heart disease, and other health issues. They are also useful for informing public policy decisions related to air quality and for helping individuals make informed choices about their daily activities. Advances in technology have made air quality monitoring systems more accurate, reliable, and accessible, making it easier than ever to track and respond to air pollution.

Keywords—Sensors, IoT, ESP32

I. INTRODUCTION (*HEADING 1*)

The quality of air in our homes, schools, and workplaces directly impacts our health, productivity, and comfort. The project aims to provide real-time information on indoor air quality through an Internet of Things (IoT) device. The primary reason for the project is to provide a solution to indoor air pollution. Indoor air pollution is caused by various sources such as cooking and heating, cleaning products, tobacco smoke, and mold. The effects of air pollution can range from mild symptoms such as headaches and dizziness to more severe effects such as asthma, lung cancer, and heart disease. It is therefore essential to monitor and control indoor air quality. The Indoor Air Quality Monitoring System is designed to monitor the air quality in real-time using various sensors. The sensors detect various pollutants such as carbon dioxide, carbon monoxide, particulate matter, and volatile organic compounds. The system is connected to the internet, enabling users to access real-time data on their mobile devices or computers. The data collected by the system is analysed to provide insights and recommendations for improving air quality.

II. PROJECT OBJECTIVE

A. Purpose

The Indoor Air Quality Monitoring System is designed to monitor the air quality in real-time using various sensors. The project's IoT aspect provides several benefits to users, such as convenience, accuracy, and cost-effectiveness. IoT technology enables the system to transmit and receive data through the internet, allowing users to monitor air quality remotely. The system is also accurate, providing precise data on air quality parameters. Additionally, IoT technology makes the system cost-effective by reducing the need for manual monitoring and analysis. Some benefits to air quality monitoring are:

- The data collected from air quality monitoring helps us assess impacts caused by poor air quality on public health.
- Air quality data helps us determine if an area is meeting the air quality standards devised by CPCB, WHO or OSHA.
- The data collected from air quality monitoring would primarily help us identify polluted areas, the level of pollution and air quality level.
- Air quality monitoring would assist in determining if air pollution control programmes devised in a locality are working efficiently or not.
- Air quality data helps us understand the mortality rate of any location due to air pollution. We can also evaluate and compare the short term and long term diseases/disorders which are a result of air pollution.
- Based upon the data collected control measures can be devised for protection of environment and health of all living organisms.

B. Rationale for Software and Hardware

Sensors are the hardware components that measure the concentration of pollutants in the air, such as particulate matter, carbon monoxide, and ozone. Different sensors are required for different types of pollutants, and they can be connected to the microcontroller through different communication protocols such as I2C, UART, or SPI. The selection of sensors depends on the target pollutants and the accuracy requirements of the system. The microcontroller is the hardware component that receives and processes data from the sensors. In an IoT architecture, the microcontroller is also responsible for transmitting data to a cloud-based data processing and visualization platform through wireless or wired communication protocols. The data processing and reporting component of the air quality monitoring system collects, processes, and presents data from the microcontroller. A cloud-based platform, Amazon Web Services (AWS) is used to host this component .The data

Identify applicable funding agency here. If none, delete this text box.

processing component of the platform involves data filtering, aggregation and reporting to the signed in user is performed to provide meaningful insights on air quality over time. The visualization component can be a web interface or a mobile application that displays the processed data in real-time or near-real-time, enabling stakeholders to make informed decisions and take appropriate actions. The rationale behind the hardware and software architecture of an air quality monitoring system is to provide a scalable, accurate, and efficient solution for collecting, processing, and presenting air quality data. This enables decision-makers to identify sources of pollution, assess the effectiveness of policies and interventions, and take timely action to protect public health.

C. *Related research to solve the problem*

The steps in our research process is as follows:

- Figuring out the need for an air quality monitoring system and what we would use to build our system.
- Researching about the pollutants we would detect using sensors. We agreed to use detectors and measure the levels of carbon monoxide, hydrogen, LPG, particulate matter, temperature and humidity.
- Investigated the most affordable and accurate sensors we could use, which were the PM 2.5 sensor for calibrating particulate matter, the MQ sensor for LPG, carbon monoxide, and hydrogen detection and finally the AHT20 for measuring temperature and humidity.
- Within the scope of the project, we used ESP32 microcontroller to program our sensors to detect and signal when threshold values exceed.
- In the next few steps, we researched how to connect a humidifier to work with the AHT20 sensor to be able to control it via the AWS ec2 hosted website.
- After we had all the sensors we needed, we divided the work of building the server and calibrating the sensors amongst ourselves.
- After finishing the above step, we worked on connecting the humidifier to our system of sensors. Though initially we used a TP-Link smart plug to do so, we were instructed by Prof. Amr Elkady that it was beyond our scope and we had to use a Motor shield to do it instead

III. DESIGN

This project is designed as an air 'quality sensor system', where different air quality sensors such as MQ2 (Gas sensor), PM2.5 (Particulate matter sensor), AHT20 (Temperature & Humidity sensor) are incorporated and controller by ESP32 (System on Chip - microcontroller). The data collected by these sensors is then sent to a remote server which processes and stores this information. This Information is then utilised by the server to either send notifications to the admin user or commands to switch ON/OFF other external devices like humidifiers. The following block diagram gives the high-level architecture of this system.

The Metrics that we collect data on are:

Gases (smoke, CO, Propane, LPG, Alcohol, Hydrogen).

Particulate Matter.

Humidity and Temperature.

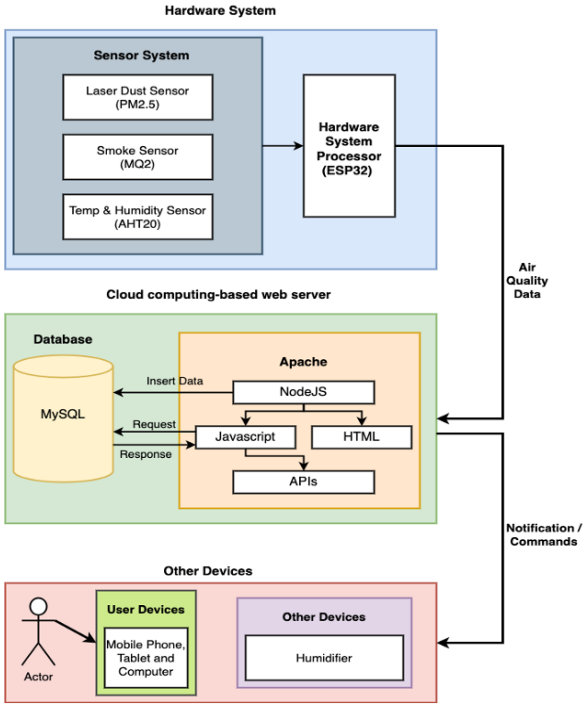

Fig 1. Block Diagram

One of the major challenges that we faced during this project was of controlling the humidifier from the server, initially we used a smart plug to controller the humidifier and we were able to control the smart plug from the server but upon further deliberation and our professor's insight we understood that using an over the counter smart plug defies the whole purpose of our project and then we moved to creating a smart plug like system of our own using ESP32, a relay switch and a voltage boost converter board.

A. *Hardware Design*

The hardware system:

1. ESP32: It's the main hardware module used in this project, we have used 3 ESP32 chips to power and control the sensor system. ESP32 is a microcontroller unit (MCU) with high level of integration, low power consumption and powerful CPU. We have used its Station Mode (STA) Wi-Fi connectivity property to connect to our cloud-based web server, which it then uses to post the collected sensor data on to the server. The following chip diagram gives the high-level architecture of ESP32.

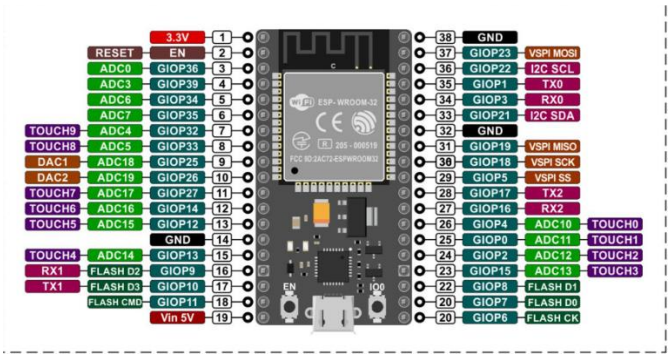

Fig 2. ESP32 Block Diagram

2. PM2.5 Sensor: We have used This sensor to find the concentration of particulate matter in the air, this sensor uses the scattering of light to find the concentration of PM 2.5 in the given air sample.

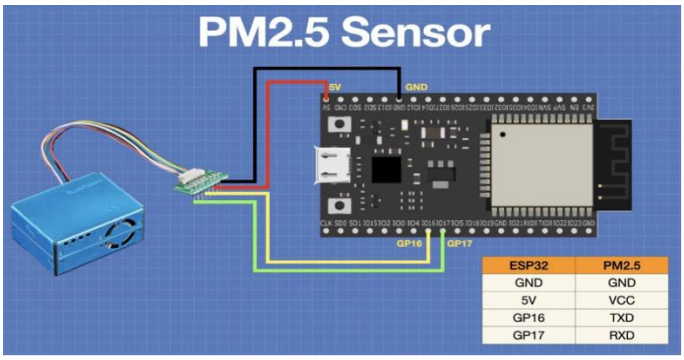

Fig3. PM 2.5 connected to ESP32

3. MQ2 Sensor: We have used an MQ2 gas sensor to sense the concentration of gases like LPG, Propane, Hydrogen, CO (Carbon Monoxide), Alcohol and smoke in the air. This sensor helps us detect whether the environment around us has harmful levels of gas or not. We have also included a buzzer into this assembly which sounds an alarm when the smoke level in the surrounding goes beyond a certain threshold.

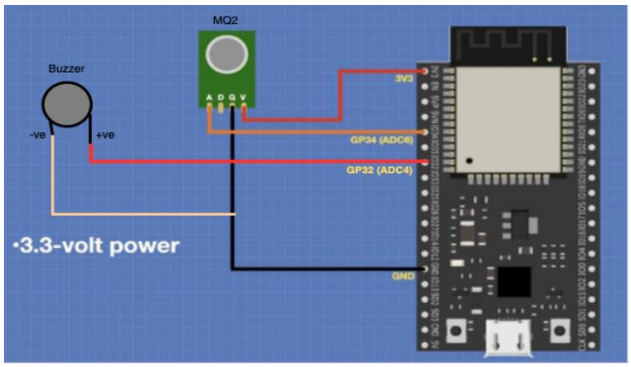

Fig 4. MQ2 Sensor connected to ESP32

4. AHT20 Sensor: This is a precise Temperature and Humidity sensing device; we have used the humidity data collected by this sensor to switch ON/OFF an external device (humidifier) directly from the server and to also send notification to the user if the humidity concentration goes below or above the threshold value of 30% and 50% respectively.

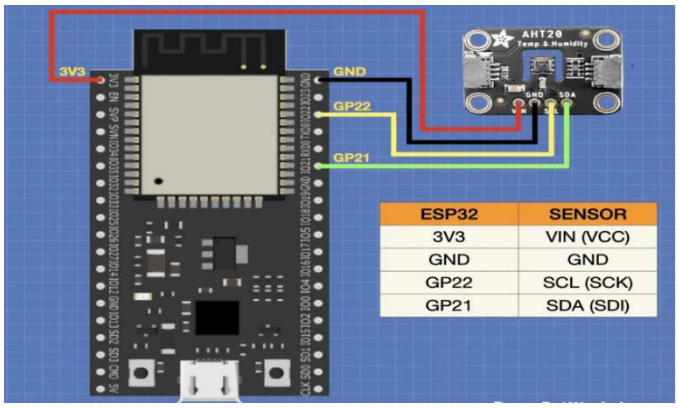

Fig 5. AHT20 connected to ESP32

5. Humidifier connected through ESP32: To control the humidifier from the server we have interfaced the humidifier to the ESP32 with the help of a relay switch and a voltage boost converter board. This was done to provide the humidifier with adequate supply of power and voltage.

   Relay Switch: A relay switch is an electrically operated switch that uses electromagnetism to convert small electrical stimuli into larger currents.

   Voltage boost converter board: A boost converter (step-up converter) is a DC-to-DC power converter that steps up voltage.

   The following figure shows the whole device system and its connections, wherein the sensors are connected to their respective ESP32 modules to collect and process their data.

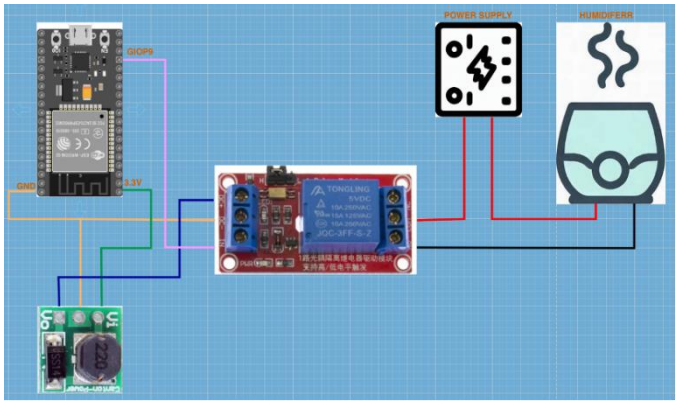

Fig 6. Humidifier Connected to ESP32

The following figure shows the whole device system and its connections, wherein the sensors are connected to their respective ESP32 modules to collect and process their data.

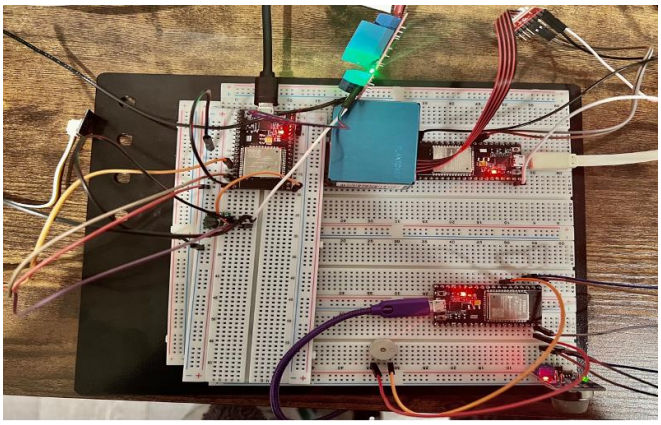

Fig 7. Hardware sensor system

B. Software Architechture

1. Firmware Design and Implementation:

Our System firmware is written using Arduino integrated development environment (IDE) and is uploaded into the ESP32 board. The program is written in C/C++ programming language and is compiled into machine code before being uploaded to the board, this firmware is then executed by the microcontroller which then uses it to process the data collected by the sensors and send it to the server. This firmware comprises three program modules, one each for every sensor system.

Program files and the libraries used in them:

Program Modules	**Libraries Used**
MQ_2_ESP32.ino: To process MQ2 sensor's data and send it to the server.	**MQUnifiedsensor:** An Arduino library to read the MQ2 sensor data easily. **WiFi:** To connect to the internet, and it uses an Arduino WiFi shield to do so. **HTTPClient:** To make http post requests to the server to post sensor data.
ParticleSensor.ino: To process the PM2.5 sensor's data and send it to the server.	**WiFi:** To connect to the internet, and it uses an Arduino WiFi shield to do so. **HTTPClient:** To make http post requests to the server to post sensor data.
TempAndHumidity.ino: To process AHT20 sensor's data and send it to the server.	**Adafruit_AHTX0:** To read temperature and humidity data from AHT20 sensor. **Wire:** To communicate with I2C devices and AHT20 sensor is one of them. **WiFi:** To connect to the internet, and it uses Arduino WiFi shield to do so. **HTTPClient:** To make http post requests to the server to post sensor data.

TABLE I. List of modules and libraries used

2. Front End Design and Implementation:

Our front-end design is quite simple and uses HTML, CSS, JavaScript, and Ajax - HTML which are some essential tools for developing modern web applications. HTML + CSS is used to structure and display content on web pages like signup, login, and dashboard page, while JavaScript is used to add interactivity and dynamic behaviour. And Ajax is used to make http requests to the server and handle the response.

HTML components of our front-end design:

HTML Component	**Functionality**
Login	This is the first html that is rendered for the user to login.
Register	This html provides the user components for registering the new us er in our system.
Dashboard	This is the view that the users get once the user is logged in. Here it contains four command buttons and two tables. Commands - Reload, Send Report, Turn Device On, Turn Device Off.
Table	This the html template which is used in the email that is send to the user

TABLE II: List of HTML component

3. Backend Design and Implementation:

We have used Nginx as the web server. It takes the incoming request from users and redirects them to the appropriate back-end server. Our application is Flask based, and we have used Gunicorn as the Python Web Server Gateway Interface (WSGI) HTTP server that is commonly used with Flask applications. It allows a way to handle incoming requests and manage multiple instances of the flask application. Python Flask is a lightweight web framework that allows users to create web applications quickly and easily. It includes features like URL routing, template rendering, and request handling. This is where we have defined our logic of handling users and sensor data requests. Our application also integrates with MySQL; thus, we have used MySQL connector for our python application to interact with MySQL for reading and storing data. We also render the html files that will be sent back as an email to the user here. These files are basically the HTML of the email notification that gets sent to the user when any deviation in sensor data is detected or when the user asks for a report.

Components used on our server:

Components	**Functionality**
flask	Used to perform API routing and rendering of html pages.
flask_mail	Used to send sensor data reports to the users and admin.
smtp_mail_setup	We have created a user with google named 'iot315coen@gmail.com' and enabled the application credentials for this user so that these credentials can be used by our server app to send the emails.

mysql.connector	Used to interact with the MySQL database, reading sensor data, and committing the sensor data into the database.
session	We have also used session provided by flask to keep track of logged in user

TABLE III: Server Component

4. Exposed API List:

On our server we support these Rest APIs:

APIs Supported	**Methods Included**
/register	Used by the UX to send the credentials to register the user.
/login	Used by the UX to send the credentials to register the user.
/sensordata	GET: Returns the most recent sensor data that is sent by the devices. POST: Used by the devices to send sensor data to the server and server commits the sensor data into the MySQL database. This also has the logic to toggle the humidifier device status on the server depending on the humidity metric value.
/report	POST: used by the UX to send the command to the server to generate and send reports on the sensor data to the currently logged in user and admin via email
/devicestatus	Used by the ESP32 to toggle the humidifier device power state. ON/OFF.
/deviceon	POST: used by the UX to send the command to the server to switch on the humidifier device.
/deviceoff	POST: Used by the UX to send the command to the server to switch off the humidifier device.

TABLE IV. APIs

C. Database Scheme

MySQL is a popular relational database management system that is often used for web applications. It provides a way to store and retrieve data efficiently and can scale to handle large amounts of data. MySQL is compatible with many programming languages, including Python, and integrates well with Flask applications.

We have used MySQL to create a Database named 'iotCOEN315' and this database consists of two tables. Users and SensorData, users' tables contain information for registered users, and SensorData contains information for the incoming data on various metrics from several sensors.

Future scope can be adding another entity to the schema for devices and a one-to-many relationship between the users and devices entities. We have not done it this time as we were only connecting with one device.

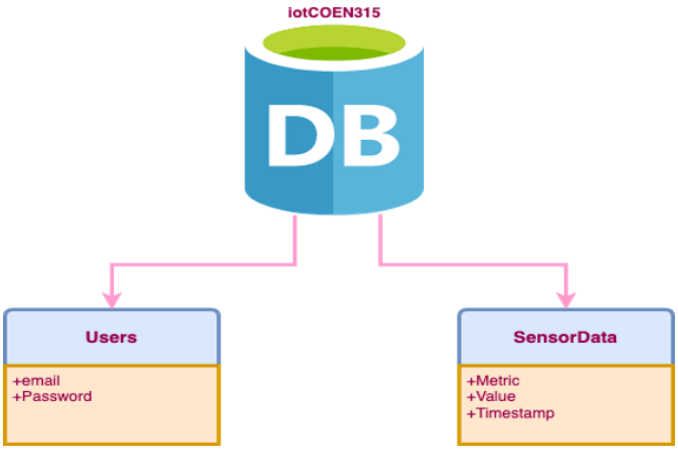

Fig 8. Database Schema

D. User Interface

Our user Interface comprises of three main sections:

1. User Registration: Our user registration page is a simple HTML form with JavaScript simple which takes in the user's email and password and a confirm password fields as inputs to register a user.

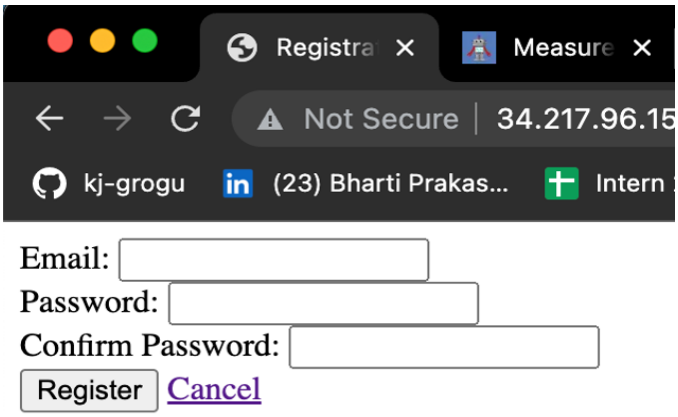

Fig 9. User Registration Page

2. User Login: Our user login page is also a simple HTML form with JavaScript simple which takes user's email and password to login an user

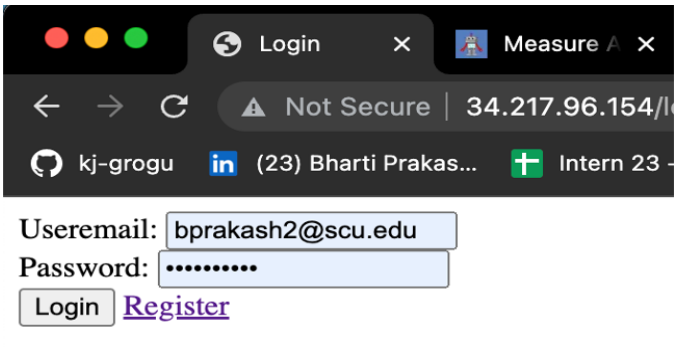

Fig 10. User Login Page

3. Dashboard: Once the user is logged in, he/she is redirected to our dashboard where all the data collected by the sensors is displayed in two tables. First table aggregates the gas, smoke, temperature, humidity, and particulate matter concentration metrics values collected in the past 24 hrs., and gives their max, min, and average values. The second table gives the most recent data points for the same metrics.

It also includes four buttons with following functionality:

Buttons	**Functionality**
Reload Data	To reload the data of the two tables
Send Report	To send a report to the logged in user on the metrics and their aggregates.
Turn ON Device	To Explicitly turn ON the humidifier from the server.
Turn OFF Device	To Explicitly turn OFF the humidifier from the server.

TABLE V. Functions

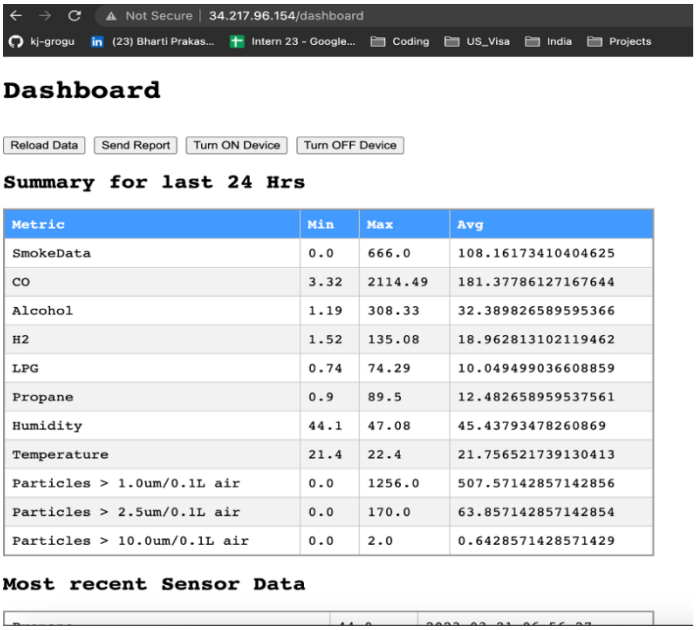

Fig 11. Server Dashboard

IV. Conclusion

To sum up, the indoor air quality monitoring system is an Internet of Things project that uses a low-cost sensor and an ESP32 microcontroller to track and analyze the air quality in enclosed environments. Through this study, we have learned the value of keeping an eye on indoor air quality as well as the potential health hazards linked to it. Also, we've learned about the many sensor types that may be used to gauge air quality and how to program the ESP32 microcontroller to gather and process sensor data.

The scope of this project is rather broad going future. A increasing demand exists for accessible and reasonably priced indoor air quality monitoring devices as air pollution and climate change continue to be significant problems.

V. References

[1] https://dronebotworkshop.com/air-quality/

VI. APPENDICES

A. *Hardware and Software Inventory:*

- https://www.amazon.com/dp/B0718T232Z?psc=1&ref=ppx_yo2ov_dt_b_product_details
- https://www.amazon.com/dp/B07838YTZW?psc=1&ref=ppx_yo2ov_dt_b_product_details
- https://www.amazon.com/dp/B09BQZP2CY?psc=1&ref=ppx_yo2ov_dt_b_product_details
- https://www.amazon.com/dp/B09KGY3S6S?psc=1&ref=ppx_yo2ov_dt_b_product_details
- https://www.amazon.com/dp/B00LW15A4W?psc=1&ref=ppx_yo2ov_dt_b_product_details
- https://www.amazon.com/dp/B07L76KLRY?psc=1&ref=ppx_yo2ov_dt_b_product_details
- https://www.amazon.com/gp/product/B07Q46XGSH/ref=sw_img_1?smid=ATVPDKIKX0DER&psc=1
- https://www.amazon.com/dp/B00LW15A4W?psc=1&ref=ppx_yo2ov_dt_b_product_details
- https://www.amazon.com/dp/B00LW15A4W?psc=1&ref=ppx_yo2ov_dt_b_product_details
- https://www.amazon.com/dp/B07L76KLRY?psc=1&ref=ppx_yo2ov_dt_b_product_details

B. Software

- • Arduino IDE v1.8.19
- • Python
- • MySQL DB
- • flask
- • HTML
- • CSS
- • javaScript
- • Other libraries (to be filled)